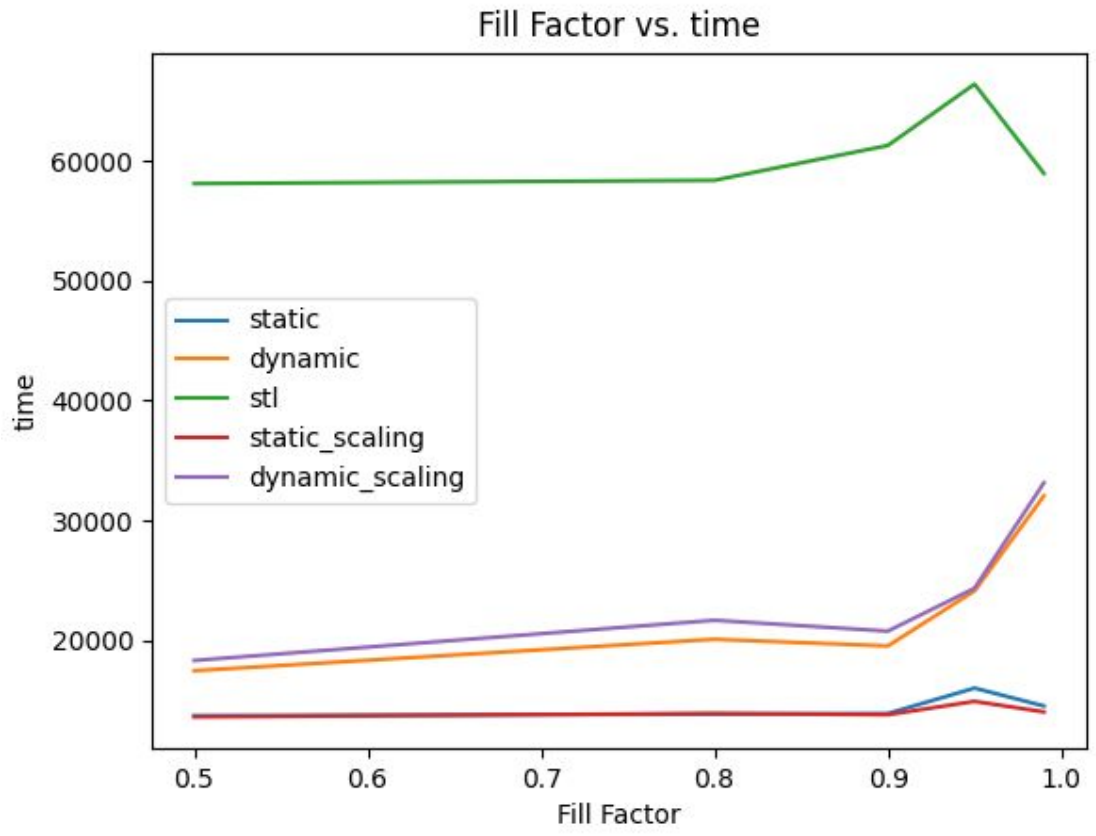
Experimentelle Algorithmik Aufgabenblatt 2 (Niklas Neesemann 609789):

Im git-repo liegen Lösungen zu 4a und b, 5a, b und c. Exercise2-Repo. Unter Evaluation liegen sowohl simex Output Directories als auch 2 Python-files (evaluation für wordcount und für hashing) und ein Jupyter Notebook. Hab Jupyter lief grade bei mir nicht, deswegen sind hier die graphischen Outputs auch nochmal zusammengefasst.

4: Möglicherweise liegt es an meiner Implementierung, aber ich kann keinen Großen Unterschied zwischen der Scaling und der Modulo Methode erkennen.



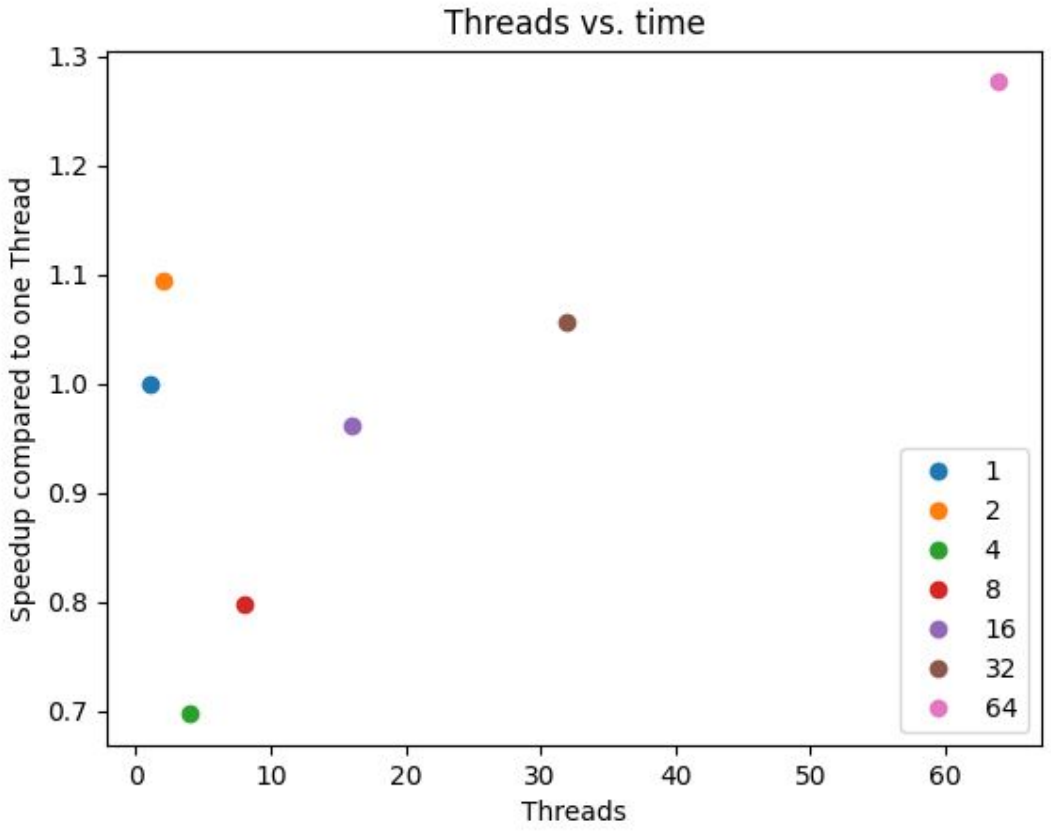
4b Perf Stat Beobachtungen: Ich glaube, dass auch hier die Ergebnisse nicht den Erwartungen entsprechen. Ich vermute stark, dass ich etwas nicht richtig implementiert habe. Mit Scaling gibt es bei mir mehr Cache Misses als mit Modulo.

Algo	MaxFill	Cache Misses
static	0.99	125393938
stl	0.99	679437053
static_scaling	0.99	116364282
dynamic_scaling	0.5	258467595
dynamic_scaling	0.8	362574195
dynamic_scaling	0.9	454307367
dynamic_scaling	0.95	604080606
dynamic_scaling	0.99	1184677265
dynamic	0.5	243637409
dynamic	0.8	359493204
dynamic	0.9	461662061
dynamic	0.95	588491145
Dynamic	0.99	1161555659

5a) allgemein: Der Algorithmus berechnet sowohl mit einem Thread als auch mit mehreren das richtige Ergebnis.

Die Messungen haben auf Gruenau1 stattgefunden.
Die Messungen für 5b und 5c wurden mit “repeat: 5” 5 mal wiederholt und für die graphische Auswertung wurde der Mittelwert der jeweils 5 Versuche genommen.

5b: Binary String Vector Size 2^24 - Speedup Factor is on the y-Axis



5c: Binary String Vector Size 2^26 - Speedup Factor is on the y-Axis